Extracción de Información Dinámica en Programación Orientada a Objetos (Java)

Hernán Bernardis[(1)], Mario Berón[(1)], Daniel Riesco[(1)], Pedro Rangel Herniques[(2)], Maria J. Pereira[(3)]

[(1)]Departamento de Informática/Facultad de Ciencias Físico-Matemáticas y Naturales/ Universidad Nacional de San Luis

Ejército de los Andes 950 – San Luis - Argentina

hernanbernardis@gmail.com, mberon@unsl.edu.ar, driesco@unsl.edu.ar

[(2)] Departamento de Informática/Universidade do Minho

Braga – Portugal

pedrorangelhenriques@gmail.com

[(3)] Departamento de Informática/Instituto Politécnico de Bragança

Bragança – Portugal

mjoao@ipb.pt

RESUMEN

La *Comprensión de Programas* (CP) es una disciplina de la Ingeniería de Software cuyo objetivo es facilitar el entendimiento de los sistemas. Para lograr esto, se vale del desarrollo de Métodos, Técnicas, Estrategias y Herramientas que permiten comprender las funcionalidades del sistema de estudio.

Uno de los principales desafíos en CP es establecer una relación entre el *Dominio del Problema* y el *Dominio del Programa*. El primero se relaciona con el comportamiento del sistema de estudio; mientras que el segundo se centra en las componentes del programa para producir dicho comportamiento. Una forma de construir esta relación consiste en elaborar una representación para cada dominio y luego establecer un procedimiento de vinculación entre ambas representaciones. La realización de la tarea previamente mencionada implica extraer información de ambos dominios, para lo cual existen múltiples técnicas.

Dentro de lo que a la extracción de información del programa se refiere, existen muchos métodos y herramientas desarrolladas, cada una de las cuales pueden ser clasificadas en base al tipo de información que extraen. Así, se tienen técnicas de extracción de información estática o dinámica. Las primeras extraen información desde el código fuente sin ejecutar el sistema. Las segundas están relacionadas con información de tiempo de ejecución.

En este artículo se describe una línea de investigación que se centra en la extracción de la información dinámica de los sistemas de software.

Palabras Claves: Comprensión de Programas, Modelos, Métodos, Técnicas, Herramientas, Extracción de Información Dinámica, Programación Orientada Objetos.

CONTEXTO

La línea de investigación descripta en este artículo se encuentra enmarcada en el contexto del proyecto: *Ingeniería del Software: Conceptos, Métodos, Técnicas y Herramientas en un Contexto de Ingeniería de Software en Evolución* de la Universidad Nacional de San Luis. Dicho proyecto, es reconocido por el programa de incentivos, y es la continuación de diferentes proyectos de investigación de gran éxito a nivel nacional e internacional.

También se forma parte del proyecto bilateral entre la Universidade do Minho (Portugal) y la Universidad Nacional de San Luis (Argentina) denominado *Quixote: Desarrollo de Modelos del Dominio del Problema para Inter-relacionar las Vistas Comportamental y Operacional de Sistemas de Software* [Quix11]. Quixote fue aprobado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación (MinCyT) [Min07] y la Fundação para a Ciência e

Tecnología (FCT) [FCT97] de Portugal. Ambos entes soportan económicamente la realización de diferentes misiones de investigación desde Argentina a Portugal y viceversa.

1. INTRODUCCIÓN

La Comprensión de Programas [Nel96, BC05, STO05, LF94, BRM10, CBHP08] es un área de la Ingeniería del Software destinada a elaborar *Modelos, Métodos, Técnicas* y *Herramientas,* basadas en procesos de aprendizaje y en procesos específicos de ingeniería, con el fin de llegar a un conocimiento profundo sobre un sistema de software. Este aprendizaje es primordial para poder establecer una relación entre el Dominio del Problema y del Programa. Es decir, la relación real entre los aspectos relacionados al comportamiento del sistema en estudio y las componentes del mismo utilizadas para producir dicho comportamiento. Para reproducir la relación entre ambos dominios, se debe construir una representación para cada uno de los dominios y luego establecer una vinculación entre las mismas. Para realizar la tarea previamente mencionada, es necesario obtener información de ambos dominios. En este contexto entran en juego un conjunto de técnicas desarrolladas, las cuales son utilizadas para este fin. Las existentes aplicadas al Dominio del Programa, se clasifican en base al tipo de información que extraen, la cual puede ser estática o dinámica. Las primeras son las más conocidas y comúnmente utilizadas en compilación de programas. Las segundas permiten obtener información durante la ejecución del programa. Estas últimas presentan una serie de ventajas de importante consideración, ellas son:

- Reducen el tamaño del espacio de búsqueda respecto a las técnicas estáticas que analizan todo el código. Esto se debe a que las técnicas dinámicas extraen información de aquellas partes del programa involucradas en una ejecución particular. Un determinado escenario[1] de ejecución es muy difícil que abarque todo el código de un sistema.
- Determinan de manera inequívoca cuales son los métodos utilizados para resolver distintos sub-problemas particulares del problema general. Esto es posible mediante la instrumentación de los métodos. Ésta técnica crea trazas de ejecución que permiten realizar un seguimiento del comportamiento del programa cuando este se ejecuta, especificando los componentes utilizados. Por ejemplo, si se desea conocer que métodos se usan en el login de un sistema, se realiza una instrumentación que informe los métodos ejecutados. Luego, al realizar la ejecución del login se obtiene la información correspondiente a los métodos involucrados.
- Detectan fallas producidas durante la ejecución que permiten descubrir y notificar errores que no se detectarían en un análisis estático, como por ejemplo el fallo de una operación con enteros.
- Proveen información del comportamiento del programa bajo distintos escenarios. Cada uno de estos escenarios produce un flujo de ejecución diferente dentro del programa, que emplea distintos componentes del mismo. Mediante la IC se pueden registrar y analizar cada uno de esos flujos y las componentes involucradas de manera de poder establecer un vínculo entre estos que ayude a establecer la relación Dominio del Problema- Dominio del Programa.

En este artículo se describe una línea de investigación que se centra en el estudio y creación de técnicas de extracción de la información desde el Dominio del Programa. Particularmente, la investigación se encuentra enfocada en aquellas técnicas que permiten

[1]Un escenario es una secuencia de entradas de usuario que disparan acciones de un sistema que llevan a un resultado observable [EKS01].

extraer información dinámica del sistema de estudio.

La organización de este artículo se expone a continuación. La sección 2 describe la línea de investigación que se lleva adelante. La sección 3 presenta los resultados obtenidos hasta el momento, junto con aquellos esperados a corto plazo. Finalmente, la sección 4 describe las tareas realizadas por los recursos humanos en formación.

2. LÍNEA DE INVESTIGACIÓN

La extracción de la información estática y dinámica es fundamental para la comprensión de programas. La primera es importante porque permite determinar la correctitud sintáctica de una construcción perteneciente al lenguaje. Este tipo de información es la utilizada por los compiladores para detectar e informar errores de tipo lexicográfico, sintáctico e incluso semántico.

La segunda se centra en el comportamiento en ejecución del programa y los componentes (métodos, variables, invocaciones, etc.) relacionados a determinados escenarios de ejecución del programa.

En un principio todas las técnicas de Comprensión de Programas se focalizaban sobre al análisis del código fuente de un sistema. Esto implicaba centrar toda la atención en métodos para la extracción de información estática. En los últimos años, se reconoció la importancia de centrar las actividades relacionadas a la Comprensión de Programas sobre el comportamiento del sistema [DGN07].

Por la razón previamente mencionada, el proyecto de investigación centró su atención en el estudio y la elaboración de estrategias para recuperar información dinámica de los sistemas de software. Esto es debido a la importancia de la misma para la creación de técnicas que permiten relacionar el Dominio del Problema con el Dominio del Programa.

Una de las estrategias de extracción de información dinámica es la Instrumentación de Código (IC). La IC consiste en insertar instrucciones dentro del código fuente del programa de modo que cuando este se ejecute, proporcione información relacionada al comportamiento del mismo durante su ejecución. Esta característica permite, entre otras cosas, conocer que partes del programa se utilizan para atacar determinados aspectos del problema que el sistema resuelve. Está demás decir que esta peculiaridad es fundamental a la hora de establecer una relación Dominio del Problema-Dominio del Programa.

Actualmente, existe un esquema de instrumentación de código definido para programas escritos en C. Dicho esquema fue realizado por miembros del proyecto y fue presentado en diferentes congresos nacionales e internacionales. Se piensa que la misma idea puede ser extendida a sistemas orientados a objetos. Por esta razón, algunos miembros del equipo de investigación se han dedicado a probar la afirmación precedente.

Básicamente, la IC se basa en la inserción de instrucciones de extracción de información, de manera cuidadosa y ordenada, en determinados puntos del código fuente. Esta inserción requiere un completo análisis de los casos posibles que pueden presentarse en el código fuente del programa, teniendo en cuenta las construcciones sintácticas que la gramática del lenguaje permite. Por lo tanto, antes de realizar una IC, es necesario tener en cuenta lo siguiente.

- Información que se desea extraer,
- Gramática del lenguaje,
- Puntos del código desde los cuales se extrae dicha información,
- Posibles instrumentaciones en base a los puntos de extracción,
- Análisis de las ventajas y desventajas de cada uno de ellos.

En un proyecto anterior se creo una estrategia que permite interconectar el Dominio del Problema con el Dominio del Programa. Dicha estrategia, denominada BORS, se aplicó a sistemas desarrollados en C. Para alcanzar su objetivo, BORS utiliza tres pasos: i) Detectar las funciones relacionadas con cada objeto del

Dominio del Problema; ii) Construir un árbol de ejecución de funciones (feTree-Function Execution Tree) y iii) Crear explicaciones que permitan relacionar las funciones ejecutadas con el problema que se pretendía resolver. En este nuevo trabajo, se pretende aplicar una técnica semejante a Lenguajes Orientados a Objetos y optimizar un modelo de representación del Dominio del Problema basado en ontologías. Esta nueva metodología permitirá sistematizar una representación de dicho dominio y facilitar la obtención de la relación con la información dinámica extraída del programa.

3. RESULTADOS OBTENIDOS Y ESPERADOS

Los resultados obtenidos hasta el momento, en lo que a instrumentación de programas orientado a objetos se refiere, son los siguientes:

- Se analizó una especificación de la gramática del lenguaje Java y se pudo concluir que es posible realizar una instrumentación mediante atributos sintetizados y heredados para la extracción dinámica de información.
- Se instrumentaron de forma genérica los métodos, iteraciones y condicionales provistos por Java [Eck00]. Por instrumentación genérica se entiende a tomar la estructura general de cada una de las construcciones mencionadas previamente e insertarle las sentencias correspondientes en lugares estratégicamente seleccionados. Por ejemplo, en el caso de los métodos, se sabe que los mismos comienzan con un tipo de retorno seguido por el nombre y la lista de parámetros de los mismos. Luego, el cuerpo de este encerrado entre llaves ({,}). Al conocer esta estructura general se puede realizar una instrumentación insertando sentencias en los puntos adecuados para extraer la información deseada. Por ejemplo, si se desea extraer información respecto a qué métodos ingresa la ejecución, se insertará una sentencia luego de la llave de apertura del cuerpo del método ({) que indique la entrada al mismo.
- Se aplicó manualmente el esquema de instrumentación mencionado en el ítem precedente a programas de ejemplo. Dicha actividad tuvo como propósito la creación de trazas de ejecución para obtener información del comportamiento del programa.
- Se estudiaron herramientas de creación de Analizadores Lexicográficos y Analizadores Sintácticos automáticos que permitan utilizar los conceptos subyacentes a las Gramáticas de Atributos. Del estudio previamente mencionado se pudo concluir que AnTLR [Ant88] es la herramienta más apropiada para implementar el esquema de instrumentación definido por un integrante del proyecto.
- Se usó AnTLR con gramáticas de ejemplo para realizar instrumentaciones de prueba.

Dentro de los resultados esperados en el corto plazo se encuentran:

- La construcción de una herramienta para obtener información dinámica de programas escritos en Java,
- Ampliar el ámbito en el cual la IC puede aplicarse. El trabajo de Berón et. al. descripto en [BHU09] se basó en el paradigma imperativo, mientras que los trabajos realizados en esta línea de investigación, buscan trasladar los conceptos al paradigma orientado a objetos.
- Demostrar que la técnica no es exclusiva a un único paradigma de lenguajes de programación,
- Crear ontologías sobre la información extraída de manera de facilitar la reconstrucción virtual de la relación real Dominio del Problema-Dominio del Programa. Se pretende crear ontologías para describir los conceptos y las relaciones pertenecientes al Dominio del Problema, con el objetivo de sistematizar una representación de éste dominio, y facilitar la identificación del mapeo de estos conceptos con secuencias de ejecución.

Todas las tareas futuras mencionadas previamente permiten percibir la importancia de la línea de investigación presentada en este trabajo para la Comprensión de Programas y la Ingeniería Inversa.

4. FORMACIÓN DE RECURSOS HUMANOS

Las tareas realizadas en el contexto de la presente línea de investigación están siendo desarrolladas como parte de trabajos de Licenciatura en Ciencias de la Computación. En el futuro se piensa generar diferentes tesis de maestría y doctorado a partir de los resultados obtenidos de las tesis de licenciatura en curso. Es importante mencionar que profesores portugueses miembros del proyecto realizaron una visita a la Universidad Nacional de San Luis en el marco del proyecto Quixote. En dicha visita, los profesores portugueses brindaron una conferencia a los integrantes del proyecto en donde se expresaron ideas y conceptos que contribuyeron al desarrollo de las tesis de licenciatura antes mencionadas.

5. REFERENCIAS

[BC05] Bliman, S.; Cockburn A. Program Comprehension: Investigating the Effects of Naming Style and Documentation. Sixth Australasian User Interface Conference (AUIC 2005). Newcastle, Australia. 2005.

[Sto05] Storey, M. A. Theories, Methods and Tools in Program Comprehension: Past, Present and Future. XIII International Workshop on Program Comprehension (IWPC 2005). St Louis, USA. 2005.

[BRM10] Berón, M.; Riesco, D.; Montejano, G. Estrategias para Facilitar la Comprensión de Programas. XII Workshop de Investigadores en Cs de la Computación. El Calafate. 2010.

[LF94] Lieberman, H.; Fry, C. Bridging the Gulf Between Code and Behavior in Programming. ACM Conference on Computers and Human Interface. Denver, Colorado. 1994.

[CBHP08] Cruz, D.; Berón, M.; Henriques, P.; Pereira, M. J. Strategies for Program Inspection and Visualization. International Scientific Conference on Computer Science and Engineering. Stará Lesná, Eslovaquia. 2008.

[DGN07] Denker, M.; Greevy, O.; Nierstrasz, O. Supporting Feature Analysis with Runtime Annotations. 3º International Workshop on Program Comprehension through Dynamic Analysis. Holanda. 2007.

[EKS01] Eisenbarth, T.; Koschke, R.; Simon, D. Aiding Program Comprehension by Static and Dynamic Feature Analisys. Proceedings of the International Conference on Software Maintenance, ICSM'2001. Florencia, Italia. 2001.

[Nel96] Nelson, M. A Survey of Reverse Engineering and Program Comprehension. Software Engineering Survey. 1996.

[BHU09] Berón, M.; Henriques, P.; Uzal, R. Inspección de Programas para Interconectar las Vistas Comportamental y Operacional para la Comprensión de Programas. Tesis doctoral de Universidad Nacional de San Luis (Argentina)-Universidade do Minho (Portugal). 2009.

[Quix11] Proyecto Quixote Home Page http://www3.di.uminho.pt/~gepl/QUIXOTE/. 2011.

[Eck00] Eckel, B. Thinking in Java 2º edición. Editorial Prentice Hall. ISBN 0-13-027363-5. 2000.

[Min07] Ministerio de Ciencia, Tecnología e Innovación Productiva. Home page. http://www.mincyt.gov.ar/. 2007.

[Ant88] Another Tool for Language Recognition (AnTLR). Home Page http://www.antlr.org/. 1988.

[FCT97] Fundação para a Ciência e a Tecnologia (FCT). Home Page. www.fct.mctes.pt/. 1997.